Using Semantic Web Technologies for Management Application Integration

Tilo Nitzsche[1], Jishnu Mukerji[2], Dave Reynolds[3], Elisa Kendall[4]

[1] Hewlett-Packard Company, Management Software Business, 11000 Wolfe Road, MS 4307, Cupertino, California 94041, USA,
tilo.nitzsche@hp.com
[2] Hewlett-Packard Company, Management Software Business, 1001 Frontier Road, Suite 300, Bridgewater, New Jersey 08807, USA,
jishnu@hp.com
[3] Hewlett-Packard Company, HP Laboratories, Filton Rd., MS T54, Stoke Gifford, Bristol BS34 8QZ, United Kingdom,
dave.reynolds@hp.com
[4] Sandpiper Software, 2053 Grant Road #162, Los Altos, California 94024, USA,
ekendall@sandsoft.com

Abstract. Management Application Integration is a software engineering discipline aimed at dynamically composing, distributing, monitoring, and managing the various applications and services deployed in today's complex enterprise. Traditional approaches to integration capabilities, in particular, are relatively static, inflexible, and do not provide the level of adaptability required for emerging dynamic and increasingly granular Service Oriented Architecture (SOA) environments. In contrast to traditional integration approaches, this paper describes current research in the use of Semantic Web technologies for model exchange between management systems from HP's OpenView management product suite.

Introduction

Management Application Integration (MAI)

Management Application Integration (MAI) is a software engineering discipline aimed at dynamically composing, distributing, monitoring, and managing the various applications and services deployed in today's complex enterprise. Integration of distinct management applications is becoming increasingly common as IT environments grow more and more complex and interdependent. As the management applications themselves grow in terms of feature sets and scope, more and more of the enterprise information they manage is replicated across them. This redundancy across management applications can be difficult to detect, and even more challenging to correlate, as each application maintains a unique model of the environment.

Business managers want better visibility into their IT infrastructure and, in particular, to be able to understand and reduce the impact of IT outages. In order to achieve these goals, integration across management applications is required – not just among applications provided by one vendor, such as HP, but across applications from multiple management software vendors, since many customers have a heterogeneous set of applications and no single vendor addresses all of their requirements.

Traditional approaches to integration are relatively static, inflexible, and do not provide the level of adaptability required for emerging dynamic and more finely granular Service Oriented Architecture (SOA) environments. Today, integration between different management applications is generally accomplished in a point-to-point fashion. The applications are tightly coupled and limited to the capabilities and flexibility provided by specific APIs used for the integration. There is no common API across OpenView applications today, for example. A mix of C-based, Java-based, WMI-based, XML-based (using a number of different transport protocols), and Web Services-based APIs [1] are provided. There is little re-use possible when developing a new integration involving distinct management applications.

One effort within HP to improve upon this situation is to provide a common data model and type system across applications. The relevant metadata is stored in Object Server which acts as an object-relational database. All applications must agree on a common type system. While agreement on a common metadata vocabulary may be achievable within a single vendor management application infrastructure, in the absence of standards in this area it has been difficult to achieve in multi-vendor environments.

Another trend – not just inside HP – is using the model of Service Oriented Architecture, often using Web Services as a basis. In the management arena, there are two competing suites of Web Services-based specifications targeted at management. The first one is WSDM (Web Services Distributed Management) being standardized by OASIS [2]. The second suite of standards is centered on WS-Management [3] being standardized by DMTF. These Web Services management specifications do not cover the underlying model of what is managed They only attempt to standardize APIs and protocols for information interchange.

Outline of the paper

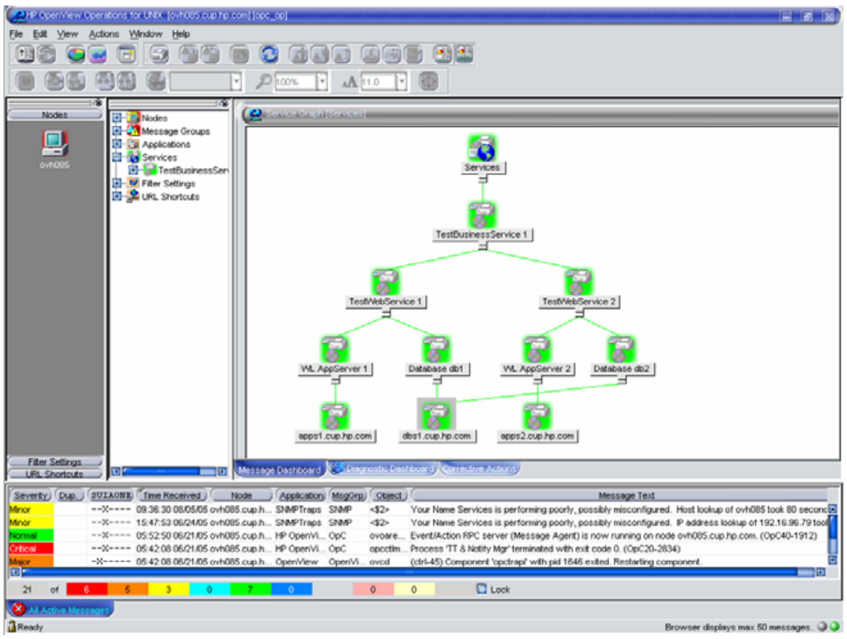
In this paper we describe ongoing research and development of a proof-of-concept implementation which uses semantic web technologies to integrate data between two different OpenView management components. This proof-of-concept was designed to enable us to compare the benefits of applying a semantic web based approach with an existing integration approach in the management space.

We begin by introducing the two components to be integrated – OpenView Operations/Service Navigator and SOA Manager. We then outline the current integration approach, the integration requirements and the semantic web based approach adopted for this study. We then describe the proof-of-concept implementation. Finally, we draw some conclusions from this initial study and discuss next steps from a research perspective.

Project Description and Goals

HP OpenView Operations and Service Navigator

HP OpenView Operations (OVO) is a distributed management application that offers management of networks, systems, databases, applications and internet services. Of primary concern for the integration with SOA Manager are the message and Service Navigator capabilities. OVO can collect events (also called OVO messages) via a number of different mechanisms. These OVO messages can be filtered, correlated or certain actions can be performed upon receiving events. Typically, messages which are not automatically handled are assigned to a certain operator who is responsible for handling these messages. Each message has a severity associated with it.



Severity	Dup.	SUIAONE	Time Received	Node	Application	MsgGrp	Object	Message Text
Minor		--X----	09:36:30 08/05/05	ovh085.cup.h...	SNMPTraps	SNMP	<$2>	Your Name Services is performing poorly, possibly misconfigured. Host lookup of ovh085 took 80 seconc
Minor		--X----	15:47:53 06/24/05	ovh085.cup.h...	SNMPTraps	SNMP	<$2>	Your Name Services is performing poorly, possibly misconfigured. IP address lookup of 192.16.96.79 too
Normal		--X----	05:52:50 06/21/05	ovh085.cup.h...	HP OpenVi...	OpC	ovoare...	Event/Action RPC server (Message Agent) is now running on node ovh085.cup.hp.com. (OpC40-1912)
Critical		--X----	05:42:08 06/21/05	ovh085.cup.h...	HP OpenVi...	OpC	opcctlm...	Process 'TT & Notify Mgr' terminated with exit code 0. (OpC20-2834)
Major		-X-----	05:42:08 06/21/05	ovh085.cup.h...	OpenView	OpenVi...	ovcd	(ctrl-45) Component 'opctrapi' with pid 1646 exited. Restarting component.

21 of 6 5 3 0 7 0 0 0

Fig. 1. Service Navigator Screenshot

Service Navigator is a component of OVO which facilitates building a hierarchical view of a managed environment. This hierarchical view is called a service map (as shown in the screenshot above). The nodes in the service map are called services. The semantics of the service map dictate that a parent node depends on a child node. Thus if a child node is disabled, the parent node will be effected (how it is effected depends on a set of configurable status propagation rules).

OVO Messages can be linked to services by various means. The status of a service will reflect the highest severity of any message associated with the service that has not been acknowledged. By default, if any child service has a higher severity, the parent will inherits that severity.

SOA Manager

HP SOA Manager [4] is a management application that provides management of Service-Oriented Architectures that are based on Web Services. It provides management functions for Web Services and captures and manages metadata on how these Web Services are provided, consumed and depend on each other.

SOA Manager is based on a central management service (called Network Services) that collects management data via a set of agents that instrument Web Services containers with management capabilities and brokers that can intercept the Web Services message flow. Originally SOA Manager was based on the Web Services Management Framework (WSMF) [5] which has been superseded by the OASIS WSDM suite of standards. WSDM provides two sets of standards, MUWS (Management Using Web Services) and MOWS (Management Of Web Services). SOA Manager currently offers a part of its functionality via MUWS and MOWS. Migration towards full support for these standards is under way.

Network Services exposes Web Services Interfaces providing access to the Web Services management information collected by agents and brokers as well as the metadata that was captured about the environment. Thus, other management applications, including 3rd party applications, can access information via Web Services interfaces that are based on standards. The model is exposed as a set of fairly fine-grained Managed Objects (MO), each being accessed as a Web Service. Each instance is exposed as one MO (*e.g.* for managed Web Services there is one MO for each Web Service). Linkages between different MOs are established using the concept of relationships. A relationship consists of a relationship type, a source and a target (it is directional). If an application wants to discover the model, it typically starts with a set of root MOs and then walks the relationships to discover everything. While this works very well for small models, there are significant performance issues for large models. The model is only available while SOA Manager is running, there is no external representation of the model that someone could look at or that could be stored somewhere. Keeping track of the model history is an important requirement for certain customers since they are required to keep an audit trail of their configuration changes.

Integrating SOA Manager with OVO/Service Navigator

The goal of the proof-of-concept project was to integrate these two components so that Service Navigator could view the set of Managed Objects (MOs) known to SOA Manager together with their dependences (see figure 1) and status information.

Current Integration

The current integration between SOA Manager and Service Navigator is based on a WSMF plug-in for Service Navigator [5]. This plug-in takes a set of root WSMF MOs as a starting point and then discovers the model following the relationships to other MOs. The service map is built based on this information. Each MO corresponds to a node in the service map. The node hierarchy is then established based on the relationship tree. The model that is displayed in Service Navigator is fixed, established based on the relationship type.

This integration approach suffers from three main disadvantages. Performance is poor for large models because several SOAP [6] requests to each MO are required to effect the model transfer. There is no external representation of the model which could be used for other purposes. Every consumer of the model receives the same view, whereas different consumers might in practice be interested in different abstractions.

An alternative integration approach is to adopt a model exchange pattern in which there is a defined, externalizable representation for the whole model which consumers can query and process. The interface must allow the consumer to not just retrieve a model but to keep it synchronized if the model changes. A key factor in adopting such an interface style is what modeling and representation technologies are used for the model exchange.

Choice of modeling technology

Three primary options were considered for the next generation implementation of the interface: DMTF CIM (Common Information Model) [26], XML and Semantic Web technologies.

CIM is based on an object-oriented view of the world [7]. It provides a number of specifications, including specifications for the meta-model and models for specific domains. Models can be externalized and exchanged as documents using the DMTF MOF (Management Object Format)[27]. CIM standards have been around for many years and CIM-based systems are broadly deployed. Unfortunately there are few standardized tools supporting CIM. One of the major user of CIM is Microsoft with WMI (Windows Management Instrumentation). Windows management is completely based on WMI, however, the APIs and protocols are proprietary. A standard called ‘CIM Operations over HTTP’ (CIM-XML) provides an XML-based protocol (which uses HTTP as transport layer) exists, but is not supported by Microsoft. HP supports both CIM and CIM-XML on all its server platforms. Other major platform vendors like IBM and Sun also support these on many of their platform products. Unfortunately the existing implementations suffer from interoperability problems among platforms from different vendors. The protocol could partially support our goals for efficient model access, but efficient usage is difficult to achieve.

XML could also be used to represent models [8]. While higher-level abstractions are provided in XML Schema that allow expressing things like a type hierarchies, tool-support (*e.g.* for XPath/XSLT) is very limited. XML does not provide support for lateral associations directly (*i.e.*, between different model components, if a model can not be expressed entirely as a hierarchy). Thus, a user-defined mechanism is necessary, at least in part. When multiple data providers exist and XML documents need to be merged, the merging process needs to be specifically implemented. This process needs to take into account the identity of the things described in the XML documents, which, for a given ‘object’, may be distributed across multiple documents.

Researchers in knowledge representation and the Semantic Web have been defining tools and methodologies that make it possible to formally capture semantic knowledge [9, 10]. Once captured, additional automation can leverage this knowledge to streamline the alignment of information models that is a prerequisite to information exchange among multiple parties. A Semantic Web approach provides support for higher-level abstractions that are not available in basic XML, including mechanisms for expressing complex relationships among types, set based intersections and unions, lattice relations among concepts, and the ability to indicate whether two model elements are the same or different from one another [11,12]. These capabilities are important for identity, model comparison, alignment, and merging, particularly across vendor implementations. Exchange of named ontologies (or named RDF graphs) would also allow the exchange of larger models as single entities, and can facilitate deeper understanding of model patterns (such as recognizing events that consist of certain patterns) as well as management of additional metadata about the models themselves [13]. This approach also enables much more flexibility from a modeling perspective, including the ability to exchange models that change dynamically based on managed application state, for example, as well as scalability, since models might be generated automatically rather than representing variants of a predetermined set of standard models. Tool support, particularly for model comparison, alignment, and merging, is limited, but there is increasing support in the community for these technologies. HP Labs has one of the most mature tool sets for use as a basis for this work as well as a team of recognized experts in the field [14]. Accessibility to this team was a key factor in the decision-making process.

RDF and OWL allow easy merging of data from multiple sources. Multiple sets of RDF statements can simply be concatenated. The determination if statements refer to the same entity can be automatically made by an OWL reasoner, if the ontology is sufficiently rich. In some management applications a consistent URI naming scheme can be adopted to enable trivial model merging. In other cases properties of the management objects can be used to establish identity via OWL InverseFunctionalProperty declarations.

Synergy within HP for an MDA® / Semantic Web Approach

In addition to the issues discussed above with respect to identity, model comparison, and so forth, critical requirements for long-term product planning included:

- Ease of use and limited potential learning curve for customers
- Interoperability within a broader enterprise framework
- Scalability, flexibility, and extensibility of the interface and metadata models embodied in the ontology components delivered with the product

Independently of the Semantic Web activity, there has been quite a bit of work on model-driven schema and data translation that obviates the need for detailed hand coding of syntactic transformations, and this work is now beginning to be combined synergistically with semantic alignment techniques [15, 16]. A Model Driven Architecture® based approach to integration insulates the business applications from technology evolution for increased portability and platform independence, cross platform interoperability, and still supports domain-relevant specificity, all of which are extremely important for long-term viability of MAI applications. Early indications were that the marriage of MDA with knowledge representation and reasoning technology could improve approaches to ontology and knowledge base development and assist in automating business semantics interchange and execution. We were also concerned that requiring our IT customers to adopt an unfamiliar and potentially challenging set of technologies might have a steep learning curve and less than satisfactory results. Ease of use from a customer perspective was high on our priority list, and the ability to use familiar UML tools for ontology development was critical to the decision making process. An approach that incorporates XML-based Semantic Web standards and an MDA-based methodology enables us to leverage the best of both worlds as well as existing OMG MOF[29]-based management capabilities, and addresses all of these requirements [17].

Implementation

In outline the implementation approach was to define two ontologies to represent the differing world views of the two applications. The SOA Manager data was expressed as an instance model using the SOA Manager ontology. The Service Navigator application maintains a synchronized view onto this instance model and creates a transformed view, using RDF rules, to conform to its own ontology.

Jena

The implementation is based on Jena [18], a set of open source Java libraries that allows manipulation and storage of RDF data (Jena is developed by HP Laboratories Bristol). One normally operates on an entity called Model. This Model can be based on a number of different storage mechanisms, including in-memory storage and database back-ends.

Higher-level APIs are provided for working with OWL data. Jena also includes support for OWL reasoning based primarily on its general purpose rules engine [19]. Specific subsets of reasoning support can be selected. A highly efficient implementation (not using the rules engine) that is used to compute inheritance for classes and slots is included. Performance of the full OWL inference engine was not sufficient to be usable for the SOA Manager ontology (which is OWL Full as of now).

Jena provides an event API, where an event listener can be notified whenever a model changes. The listener can access the set of RDF statements that are removed from the model and the set of statements that is added to the model.

High-Level Architecture

The integration is based on the synchronization of model repositories and is illustrated in Figure 2. The left hand side of the diagram shows SOA Manager which is using the

Jena Java API to populate an in-memory OWL model of SOA Manager with the appropriate instance data. This model is explicitly kept in sync with the internal state of SOA Manager.

A WSRF [24] model resource is subscribed to model changes via the event API that Jena provides. This model resource is responsible for providing external access to the model. It provides a method 'getModel' to get the current model state and support a WS Notification [24] 'modelChangedEvent' which contains a list of RDF statements removed from the model and a set statements added to the model since the last event. Thus a consumer can do an initial 'getModel' request and then keep synchronized with the model by subscribing to the model change events. This model synchronization mechanism is completely generic and could be used for synchronizing any model store with another remote model store.

The right hand side of the diagram shows the Service Navigator side which is using the synchronization mechanism as described above to get a synchronized copy of the SOA Manager model.

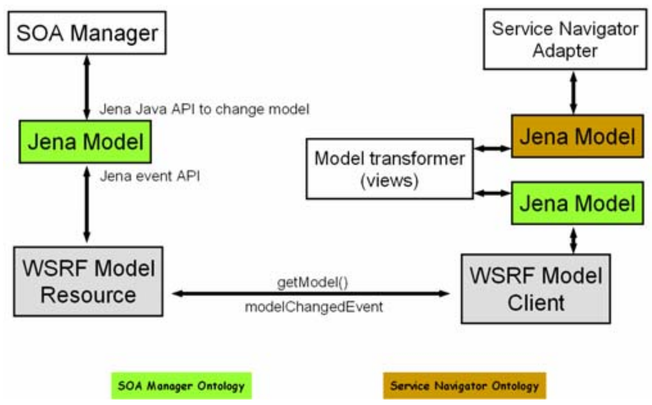


Fig. 2. Integration Architecture

Now, a model transformation is performed (described below). This transformation takes a model in the SOA Manager ontology and transforms it to the Service Navigator ontology. A Service Navigator adapter monitors the model and builds the related service map. It also monitors which status the service navigator nodes are supposed to have and creates and acknowledges OVO messages based on this.

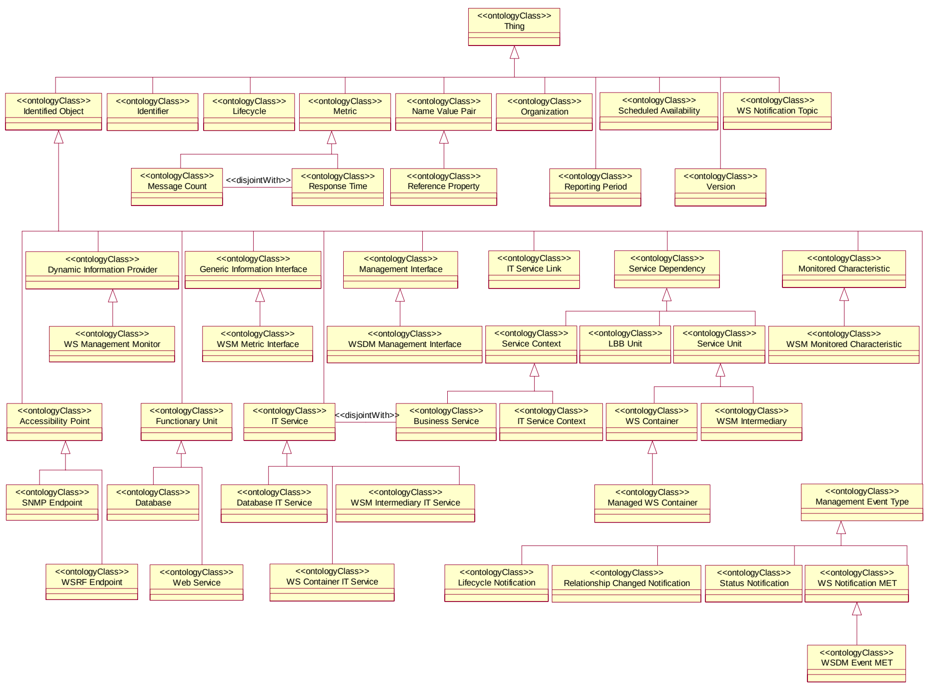
Ontology Overview

Two ontologies were developed as a part of the preliminary proof-of-concept effort: a SOA Manager ontology and a Service Navigator ontology. These were developed by independent, distributed teams with little interaction until the application was actually integrated, which assisted in validating our approach.

The SOA Manager ontology was loosely based on prior work with respect to SOA Manager architecture and WSMF implementation. It was developed using Sandpiper's Visual Ontology Modeler (VOM), an add-in to IBM Rational Rose, which supports ontology development in a UML environment [20]. The Service Navigator ontology was developed using the Protégé tool from Stanford Medical Informatics [21, 22]. Use of two distinct tools by independent teams provided an opportunity for comparing the state of the art in ontology development tools, to understand the learning curve required of HP developers and potentially of OpenView customers, and to test whether or not an MDA-based approach was feasible. This study is ongoing and beyond the intended scope of this paper.

SOA Manager Ontology

The class hierarchy of the SOA Manager ontology is given in Figure 3. It is depicted using Sandpiper's VOM tool, which implements a UML profile for RDF and OWL, a basis for the emerging Ontology Definition Metamodel (ODM) standard [23].



<<ontologyClass>> Thing
<<ontologyClass>> Identified Object
<<ontologyClass>> Identifier
<<ontologyClass>> Lifecycle
<<ontologyClass>> Metric
<<ontologyClass>> Name Value Pair
<<ontologyClass>> Organization
<<ontologyClass>> Scheduled Availability
<<ontologyClass>> WS Notification Topic
<<ontologyClass>> Message Count
<<disjointWith>>
<<ontologyClass>> Response Time
<<ontologyClass>> Reference Property
<<ontologyClass>> Reporting Period
<<ontologyClass>> Version
<<ontologyClass>> Dynamic Information Provider
<<ontologyClass>> Generic Information Interface
<<ontologyClass>> Management Interface
<<ontologyClass>> IT Service Link
<<ontologyClass>> Service Dependency
<<ontologyClass>> Monitored Characteristic
<<ontologyClass>> WS Management Monitor
<<ontologyClass>> WSM Metric Interface
<<ontologyClass>> WSDM Management Interface
<<ontologyClass>> Service Context
<<ontologyClass>> LBB Unit
<<ontologyClass>> Service Unit
<<ontologyClass>> WSM Monitored Characteristic
<<ontologyClass>> Accessibility Point
<<ontologyClass>> Functionary Unit
<<ontologyClass>> IT Service
<<disjointWith>>
<<ontologyClass>> Business Service
<<ontologyClass>> IT Service Context
<<ontologyClass>> WS Container
<<ontologyClass>> WSM Intermediary
<<ontologyClass>> SNMP Endpoint
<<ontologyClass>> Database
<<ontologyClass>> Database IT Service
<<ontologyClass>> WSM Intermediary IT Service
<<ontologyClass>> Managed WS Container
<<ontologyClass>> Management Event Type
<<ontologyClass>> WSRF Endpoint
<<ontologyClass>> Web Service
<<ontologyClass>> WS Container IT Service
<<ontologyClass>> Lifecycle Notification
<<ontologyClass>> Relationship Changed Notification
<<ontologyClass>> Status Notification
<<ontologyClass>> WS Notification MET
<<ontologyClass>> WSDM Event MET

Fig. 3. SOA Manager Ontology in VOM

The ontology defines a wide range of applicable concepts, from basic identification of managed objects to managed events, monitored characteristics, service context, and so forth.

Service Navigator Ontology

The Service Navigator ontology contains a couple of key concepts. It is centered around the class ‘SNNode’, which represents a node in the Service Navigator service map. The node hierarchy is formed using the ‘dependsOnSNNode’ property which establishes a parent-child relationship between two nodes. Nodes have a status property, which reflects the current status to be shown in Service Navigator.

Model Transformation

The model transformation from the SOA Manager ontology to the Service Navigator ontology is based on the general-purpose rules engine that Jena provides. The rules engine can operate in backward-chaining mode (new RDF statements are not explicitly generated, the rules engine is used to answer queries against the model) or forward chaining (new RDF statements for each rule are generated). The forward chaining mode can be used for model rewriting or transformation. It is possible to separately access the generated RDF model.

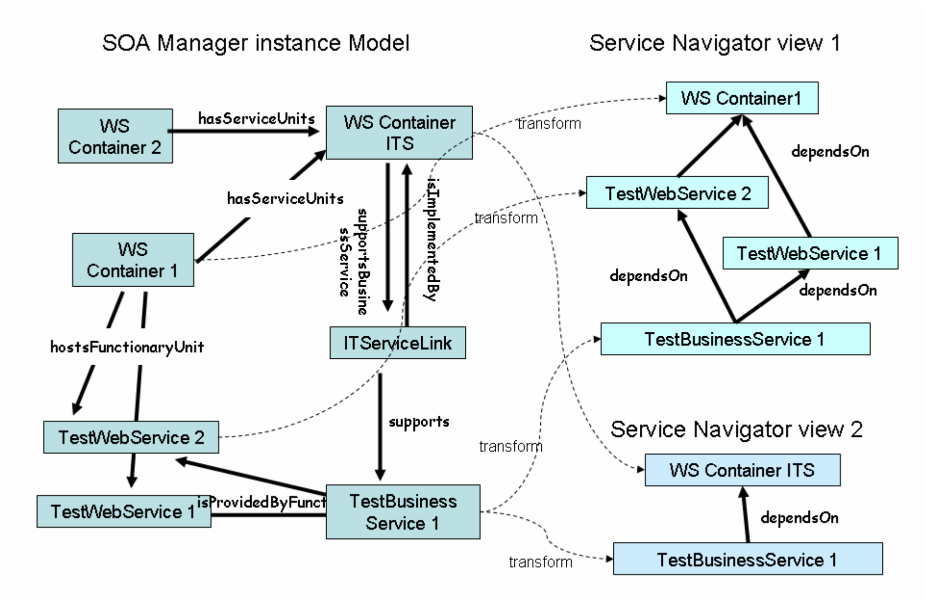


Fig. 4. Example instance model and its transforms

Figure 4 shows sample instance data in the SOA Manager (SOAm) Ontology and how it is transformed into two different models in the Service Navigator (SN) Ontology. The boxes represent instances of OWL classes, the arrows represent properties. The essence of the transformation is to select resources within the SOAm Model, based on their class, and map them into Nodes within the SN view. Properties of the SN view such as names, dependency links and status attributes are directly derivable from equivalent properties in the SOAm models. The mappings are specified using forward production rules.

Transformation Rule Set

Below is a subset of the transformation rules used for the first view in Service Navigator. There is a set of triples on the input side of a rule and a set of triples on the output side of a rule. The engine tries to match the RDF triples on the input side against the current model. Everything that starts with a question mark is an unbound variable and will be bound against data in the model by the engine. The triples on the output side are new RDF triples that will be created.

```
[rule1: (?B rdf:type soamgr:BusinessService),
        (?B soamgr:hasName ?N),
        (?B soamgr:hasUniqueIdentifier ?ID)
    ->
        (?B rdf:type sn:SNRootNode),
        (?B sn:hasSNNodeID ?ID),
        (?B sn:hasSNLabel ?N)]
```

We are looking for something of type ‘BusinessService’ to bind to the variable ‘?B’. We then select the name and uuid of the ‘BusinessService’ and bind those to ‘?N’ and ‘?ID’. On the output side, we give the ‘BusinessService’ a new type ‘SNRootNode’ and add the properties SNNodeID and SNLabel, re-using the identity of the old ‘BusinessService’ instance. We could also create a new instance.

```
[rule2: (?WS rdf:type soamgr:WebService),
        (?WS soamgr:hasUniqueIdentifier ?ID),
        (?WS soamgr:hasName ?N)
     ->
        (?WS rdf:type sn:SNNode),
        (?WS sn:hasSNNodeID ?ID),
        (?WS sn:hasSNLabel ?N)]
```

We create SNNodes for ‘WebService’ instances.

```
[rule3: (?SC
soamgr:hasServiceContextManagementInterface ?SCMI),
        (?SC rdf:type sn:SNNode),
        (?SCMI soamgr:hasMIAccessibilityPoint ?AP),
        (?AP rdf:type soamgr:WSRFEndpoint),
        (?AP soamgr:hasEndpointURL ?url)
    ->
        (?AP rdf:type sn:WSDMInterface),
        (?AP sn:hasEndpointURL ?url),
        (?SC sn:hasWSDMManagementIntf ?AP)]

[rule4: (?AP sn:hasEndpointURL ?url),
        (?AP soamgr:hasReferenceProperties ?RP),
        (?RP soamgr:hasPropertyName ?NAME),
        (?RP soamgr:hasPropertyValue ?VALUE)
    ->
        (?RP rdf:type sn:ReferenceProperty),
        (?AP sn:hasReferenceProperty ?RP),
        (?RP sn:hasRPName ?NAME),
        (?RP sn:hasRPValue ?VALUE)]
```

We copy over the WSDM management interface information. Note that these rules are recursive (e.g. in rule4 we check that we already assigned the new type SNNode).

```
[rule5: (?C rdf:type soamgr:ManagedWSContainer),
        (?C soamgr:hasName ?N),
        (?C soamgr:hasUniqueIdentifier ?ID)
     ->
        (?C rdf:type sn:SNNode),
        (?C sn:hasSNNodeID ?ID),
        (?C sn:hasSNLabel ?N)]
```

We create SNNodes for ManagedWSContainers.

```
  [rule6: (?B soamgr:isProvidedByFunctionaryUnit ?FU),
          (?B rdf:type sn:SNNode),
          (?FU rdf:type sn:SNNode)
       ->
          (?B sn:dependsOnSNNode ?FU)]
```

We establish the linkage from ‘BusinessService’ ‘SNNodes’ to ‘FunctionaryUnit’ ‘SNNodes’.

```
  [rule8: (?SU soamgr:hostsFunctionaryUnit ?FU),
          (?SU rdf:type sn:SNNode),
          (?FU rdf:type sn:SNNode)
     ->
          (?FU sn:dependsOnSNNode ?SU)]
```

We transform the connection between BusinessService and ITService that is established via the ITServiceLink class into the ‘dependsOnSNNode’ property between the corresponding ‘SNNodes’.

Results and conclusions

The resulting integration was successful and Service Navigator is able to access the SOA Manager models as required. In this section we compare the proof-of-concept model exchange approach with the baseline WSRF/WSDM-based integration described earlier.

One major benefit of the model exchange integration approach over the baseline is a substantial improvement in performance. The WSRF integration requires a large number of individual web service calls to traverse the exposed MOs and reconstruct the dependency model. With the new integration the model has been created in an externalizable form by SOA Manager and clients such as Service Navigator can retrieve the entire model in a single call. The models are then kept synchronized by notification calls which are able to batch up sets of changes to again reduce overhead.

In homogenous systems where the communicating components share common domain models, instance-model-exchange is a natural approach. It is made possible in this heterogeneous case through the use of model transformation to map between the conceptual models of the two components. This mapping could have been carried out in either component or through an intermediary service.

The effort required to develop the new integration was significantly less that the previous integration. Furthermore the previous integration only supported the Unix version of OVO and porting to the Windows version required a significant amount of effort. Porting the new integration is much cheaper requiring only an easy port of the Service Navigator Adapter (figure 2).

This ease of implementation can be attributed to the choice of semantic web technologies for the modeling. The RDF data model is well suited to the representation of dependency graphs such as those that arise in management applications. Developing the domain models for the two domains proved straightforward using either of the ontology development tools tested. Using RDF rules to transform between the two domains was simple, and writing new rules to accommodate alternative views is straightforward. A key factor here is that we were able to concentrate on the conceptual and data modeling problem and did not need to develop a serialization syntax or perform any processing at the syntax level.

This contrasts with the alternative of developing a custom model format in XML Schema. In that case we would have needed to develop application-specific representations for serializing the graph structures onto XML's inherently hierarchical model and transformation problems would have become entangled with these syntax issues. The clear separation of model syntax from conceptual modeling and transformation led to a significant reduction in development cost.

This approach provides a foundation for n-way integration. There are several management components available which offer complementary information to SOA manager. By exporting their models in a similar fashion and transforming them to the Service Navigator conceptual model we are able to integrate this information into a single view. The open world assumption behind RDF/OWL makes it easy to add additional information into the view from these other sources without requiring all of the communicating components to understand all of the properties involved. Each is able to pass through the integrated RDF model. Components that understand the additional data (either natively or through a transformation) can extract and process it. This ability to add information without any need for a schema change and having old components, transparently pass through data to new components, greatly eases the decoupling between components in such multi-way integrations. This has direct impact on the cost and speed of development.

Finally, the entire architecture of adapters plus rule-based transforms is entirely generic and we anticipate will be very easy to reuse for other integration use cases.

Potential Research Directions

First we plan to generalize the applications of the mapping architecture. The model-transformation approach we have used here to map between two domain models can also be applied for transformation between different versions of a single component. In a complex multi-service environment, such as management applications, version change management is a significant cost. We hope to explore the use of schema comparison techniques such as [25] to partially automate the analysis of version changes and the generation of the relevant transformation rules for version adaptation.

Secondly, one limitation of the current architecture is that the model synchronization is one-way. The consumer receives only a read-only copy of the supplier's externalized model. In more complex integration scenarios some form of write access is required. At this stage it is unclear if this simply requires a two-way synchronization protocol or whether a language for explicit updates to the exposed models is needed. We note that current standardization work on RDF query languages has not yet addressed the issue of model update.

Finally, in the case where were we are integrating data from multiple sources there is an issue with how to deal with conflicting information. We have not addressed this in the work so far and it remains an open problem. In the management domain individual sources have known strengths and weaknesses in their ability to detect and report on specific management properties. It may be that a relatively static trust mechanism would enable automated resolution of the most common conflicts in such a setting.

References

[1] WMI, http://www.microsoft.com/whdc/system/pnppwr/wmi/default.mspx

[2] WSDM MUWS, MOWS, http://devresource.hp.com/drc/specifications/wsdm/index.jsp

[3] WS Management, https://wiseman.dev.java.net/

[4] SOA Manager, http://devresource.hp.com/drc/resources/lcm4ws_overview/index.jsp

[5] WSMF, http://devresource.hp.com/drc/specifications/wsmf/index.jsp

[6] SOAP, http://www.w3.org/TR/soap/

[7] CIM Operations over HTTP, http://www.dmtf.org/standards/documents/WBEM/DSP200.html

[8] XML Schema, http://www.w3.org/XML/Schema

[9] Berners-Lee, Tim, Jim Hendler, and Ora Lassila, "The Semantic Web". In *Scientific American*, New York, NY, May 2001. See http://www.sciam.com/print_version.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21 for online version.

[10] The Semantic Web Activity of the World Wide Web Consortium (W3C). See http://www.w3.org/2001/sw/ for current research, links to standards and related work.

[11] Resource Description Framework (RDF): Concepts and Abstract Syntax. Graham Klyne and Jeremy J. Carroll, Editors. W3C Recommendation, 10 February 2004. Latest version is available at http://www.w3.org/TR/ rdf-concepts/.

[12] OWL Web Ontology Language Semantics and Abstract Syntax. W3C Recommendation 10 February 2004, Peter F. Patel-Schneider, Patrick Hayes, Ian Horrocks, eds. Latest version is available at http://www.w3.org/TR/ owl-semantics/.

[13] Carroll, Jeremy J.; Bizer, Christian; Hayes, Patrick; Stickler, Patrick. Named Graphs, Provenance and Trust. Available as HP Labs Technical Report. HPL-2004-57R1 http://www.hpl.hp.com/techreports/2004/HPL-2004-57R1.html

[14] HP Labs Semantic Web Research. http://www.hpl.hp.com/semweb/

[15] Frankel, David S. *Model Driven Architecture®: Applying MDA® to Enterprise Computing*, Wiley Publishing, Inc., Indianapolis, Indiana, 2003.

[16] Miller, Joaquin and Mukerji, Jishnu, editors, *MDA Guide 1.01*, Object Management Group, Needham, MA, June 2003. See http://www.omg.org/docs/omg/03-06-01.pdf for online version.

[17] David Frankel, Patrick Hayes, Elisa Kendall, and Deborah McGuinness, "The Model Driven Semantic Web." In Proceedings of the Enterprise Distributed Object Computing (EDOC) Conference 2004, Model Driven Semantic Web Workshop (MDSW2004), Monterey, California, September 2004. See http://www.sandsoft.com/edoc2004/ for this and several related papers on various components of the Ontology Definition Metamodel (ODM) Revised Submission Specification.

[18] Jena, http://jena.sourceforge.net/

[19] Jena Rules Engine, http://jena.sourceforge.net/inference/

[20] Ceccaroni, Luigi and Elisa Kendall. A Graphical Environment for Collaborative Ontology Development. In *Proceedings, The Second International Joint Conference on Autonomous Agents & Multi Agent Systems, Poster Session*, July 14-18, 2003, Melbourne, Australia. ACM Digital Library.

[21] Gennari, J. H., H. Cheng, R. B. Altman and M. Musen, Reuse, CORBA, and Knowledge-Based Systems. SMI Report SMI-97-0687, Stanford Medical Informatics, Stanford University, 1997.

[22] Q. Li, P. Shilane, N. F. Noy, M. Musen, Ontology Acquisition from On-line Knowledge Sources. SMI Report SMI-2000-0850, Stanford Medical Informatics, Stanford University, 2000.

[23] Ontology Definition Metamodel (ODM). Daniel T. Chang and Elisa F. Kendall, editors. Second Joint Revised Submission, 31 May 2005, Object Management Group. See http://www.omg.org/docs/ad/05-04-13.pdf.

[24] WSRF, http://devresource.hp.com/drc/specifications/wsrf/index.jsp

[25] Natalya F. Noy, Mark A. Munsen, Promptdiff: a fixed-point algorithm for comparing ontology versions, Proc. 18th National conference on Artificial Intelligence, p. 744-750. 2002

[26] Distributed Management Task Force (DMTF) Common Information Model (CIM) Specification, http://www.dmtf.org/standards/cim/cim_spec_v22

[27] DMTF MOF: Managed Object Format, http://www.dmtf.org/education/mof/

[28] OMG MOF, http://www.omg.org/technology/documents/formal/mof.htm

[29] UML, http://www.uml.org/